UNIX - FILE MANAGEMENT

All data in UNIX is organized into files. All files are organized into directories. These directories are organized into a tree-like structure called the filesystem.

When you work with UNIX, one way or another you spend most of your time working with files. This tutorial would teach you how to create and remove files, copy and rename them, create links to them etc.

In UNIX there are three basic types of files:

1. **Ordinary Files:** An ordinary file is a file on the system that contains data, text, or program instructions. In this tutorial, you look at working with ordinary files.

2. **Directories:** Directories store both special and ordinary files. For users familiar with Windows or Mac OS, UNIX directories are equivalent to folders.

3. **Special Files:** Some special files provide access to hardware such as hard drives, CD-ROM drives, modems, and Ethernet adapters. Other special files are similar to aliases or shortcuts and enable you to access a single file using different names.

Listing Files:

To list the files and directories stored in the current directory. Use the following command:

```
$ls
```

Here is the sample output of the above command:

```
$ls

bin        hosts  lib     res.03
ch07       hw1    pub     test_results
ch07.bak   hw2    res.01  users
docs       hw3    res.02  work
```

The command **ls** supports the **-1** option which would help you to get more information about the listed files:

```
$ls -l
total 1962188

drwxrwxr-x  2 amrood amrood      4096 Dec 25 09:59 uml
-rw-rw-r--  1 amrood amrood      5341 Dec 25 08:38 uml.jpg
drwxr-xr-x  2 amrood amrood      4096 Feb 15  2006 univ
drwxr-xr-x  2 root   root        4096 Dec  9  2007 urlspedia
-rw-r--r--  1 root   root      276480 Dec  9  2007 urlspedia.tar
drwxr-xr-x  8 root   root        4096 Nov 25  2007 usr
drwxr-xr-x  2    200    300      4096 Nov 25  2007 webthumb-1.01
-rwxr-xr-x  1 root   root        3192 Nov 25  2007 webthumb.php
-rw-rw-r--  1 amrood amrood     20480 Nov 25  2007 webthumb.tar
-rw-rw-r--  1 amrood amrood      5654 Aug  9  2007 yourfile.mid
-rw-rw-r--  1 amrood amrood    166255 Aug  9  2007 yourfile.swf
drwxr-xr-x 11 amrood amrood      4096 May 29  2007 zlib-1.2.3
$
```

Here is the information about all the listed columns:

1. First Column: represents file type and premission given on the file. Below is the description of all type of files.

2. Second Column: represents the number of memory blocks taken by the file or directory.
3. Third Column: represents owner of the file. This is the Unix user who created this file.
4. Fourth Column: represents group of the owner. Every Unux user would have an associated group.
5. Fifth Column: represents file size in bytes.
6. Sixth Column: represents date and time when this file was created or modified last time.
7. Seventh Column: represents file or directory name.

In the ls -l listing example, every file line began with a d, -, or l. These characters indicate the type of file that's listed.

Prefix	Description
-	Regular file, such as an ASCII text file, binary executable, or hard link.
b	Block special file. Block input/output device file such as a physical hard drive.
c	Character special file. Raw input/output device file such as a physical hard drive
d	Directory file that contains a listing of other files and directories.
l	Symbolic link file. Links on any regular file.
p	Named pipe. A mechanism for interprocess communications
s	Socket used for interprocess communication.

Meta Characters:

Meta characters have special meaning in Unix. For example * and **?** are metacharacters. We use * to match 0 or more characters, a question mark **?** matches with single character.

For Example:

```
$ls ch*.doc
```

Displays all the files whose name start with ch and ends with .doc:

```
ch01-1.doc   ch010.doc  ch02.doc    ch03-2.doc
ch04-1.doc   ch040.doc  ch05.doc    ch06-2.doc
ch01-2.doc ch02-1.doc c
```

Here * works as meta character which matches with any character. If you want to display all the files ending with just **.doc** then you can use following command:

```
$ls *.doc
```

Hidden Files:

An invisible file is one whose first character is the dot or period character (.). UNIX programs (including the shell) use most of these files to store configuration information.

Some common examples of hidden files include the files:

- **.profile:** the Bourne shell (sh) initialization script
- **.kshrc:** the Korn shell (ksh) initialization script
- **.cshrc:** the C shell (csh) initialization script
- **.rhosts:** the remote shell configuration file

To list invisible files, specify the -a option to ls:

```
$ ls -a

.          .profile       docs     lib     test_results
..         .rhosts        hosts    pub     users
.emacs     bin            hw1      res.01  work
.exrc      ch07           hw2      res.02
.kshrc     ch07.bak       hw3      res.03
$
```

- Single dot **.**: This represents current directory.
- Double dot **..**: This represents parent directory.

Note: I have put stars (*) just to show you the location where you would need to enter the current and new passwords otherwise at your system, it would not show you any character when you would type.

Creating Files:

You can use **vi** editor to create ordinary files on any Unix system. You simply need to give following command:

```
$ vi filename
```

Above command would open a file with the given filename. You would need to press key **i** to come into edit mode. Once you are in edit mode you can start writing your content in the file as below:

```
This is unix file....I created it for the first time.....
I'm going to save this content in this file.
```

Once you are done, do the following steps:

- Press key **esc** to come out of edit mode.
- Press two keys **Shift + ZZ** together to come out of the file completely.

Now you would have a file created with **filemame** in the current directory.

```
$ vi filename
$
```

Editing Files:

You can edit an existing file using **vi** editor. We would cover this in detail in a separate tutorial. But in short, you can open existing file as follows:

```
$ vi filename
```

Once file is opened, you can come in edit mode by pressing key **i** and then you can edit file as you like. If you want to move here and there inside a file then first you need to come out of edit mode by pressing key **esc** and then you can use following keys to move inside a file:

- **l** key to move to the right side.
- **h** key to move to the left side.
- **k** key to move up side in the file.
- **j** key to move down side in the file.

So using above keys you can position your cursor where ever you want to edit. Once you are positioned then you can use **i** key to come in edit mode. Edit the file, once you are done press **esc** and finally two keys **Shift + ZZ** together to come out of the file completely.

Display Content of a File:

You can use **cat** command to see the content of a file. Following is the simple example to see the content of above created file:

```
$ cat filename
This is unix file....I created it for the first time.....
I'm going to save this content in this file.
$
```

You can display line numbers by using **-b** option along with **cat** command as follows:

```
$ cat filename -b
1   This is unix file....I created it for the first time.....
2   I'm going to save this content in this file.
$
```

Counting Words in a File:

You can use the **wc** command to get a count of the total number of lines, words, and characters contained in a file. Following is the simple example to see the information about above created file:

```
$ wc filename
2  19 103 filename
$
```

Here is the detail of all the four columns:

1. First Column: represents total number of lines in the file.
2. Second Column: represents total number of words in the file.
3. Third Column: represents total number of bytes in the file. This is actual size of the file.
4. Fourth Column: represents file name.

You can give multiple files at a time to get the information about those file. Here is simple syntax:

```
$ wc filename1 filename2 filename3
```

Copying Files:

To make a copy of a file use the **cp** command. The basic syntax of the command is:

```
$ cp source_file destination_file
```

Following is the example to create a copy of existing file **filename**.

```
$ cp filename copyfile
$
```

Now you would find one more file **copyfile** in your current directory. This file would be exactly same as original file **filename**.

Renaming Files:

To change the name of a file use the **mv** command. Its basic syntax is:

```
$ mv old_file new_file
```

Following is the example which would rename existing file **filename** to **newfile**:

```
$ mv filename newfile
$
```

The **mv** command would move existing file completely into new file. So in this case you would fine only **newfile** in your current directory.

Deleting Files:

To delete an existing file use the **rm** command. Its basic syntax is:

```
$ rm filename
```

Caution: It may be dangerous to delete a file because it may contain useful information. So be careful while using this command. It is recommended to use **-i** option along with **rm** command.

Following is the example which would completely remove existing file **filename**:

```
$ rm filename
$
```

You can remove multiple files at a tile as follows:

```
$ rm filename1 filename2 filename3
$
```

Standard Unix Streams:

Under normal circumstances every Unix program has three streams (files) opened for it when it starts up:

1. **stdin :** This is referred to as *standard input* and associated file descriptor is 0. This is also represented as STDIN. Unix program would read default input from STDIN.

2. **stdout :** This is referred to as *standard output* and associated file descriptor is 1. This is also represented as STDOUT. Unix program would write default output at STDOUT

3. **stderr :** This is referred to as *standard error* and associated file descriptor is 2. This is also represented as

STDERR. Unix program would write all the error message at STDERR.